第3章 FPGAで動かすレトロ調ゲーム

ブロック崩しゲームの製作

江崎雅康，岩田正雄

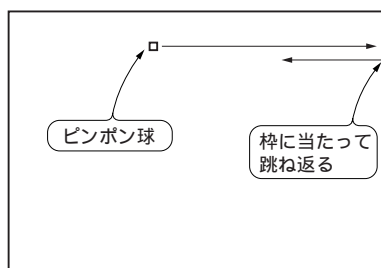
第２章のマイクロプロセッサ回路と，本誌2007年８月号特集１ 第２章のアナログRGB出力回路を使って「何か，意味のある作品を…」と思い立って，「ブロック崩しゲーム」の製作に着手した．（筆者）

1 FPGAでアーケード・ゲーム製作

黎明(れいめい)期のテレビ・ゲームの基板には100個近くのTTL標準ロジックICが並んでいました．標準ロジックICだけで，テレビの同期信号から画面上のアクションまで組み上げられていたアーケード・ゲームもありました．

当時のロジックIC 1個を100ゲートとしても，25万ゲートのSpartan-3E（XC3S250）には2,500個分のICを組み込めるパワーがあります．

FPGAであれば，多くのICをコツコツはんだ付けする手間も不要です．パソコン上でHDLコードを記述するだけで回路が完成します．30年前には考えられなかった25万ゲートのFPGAを使ってアーケード・ゲームを作ってみよう．これが開発の動機です．

図1 ピンポン玉の動作（水平方向のみ）
10×10ドットのピンポン球（正方形で表現）を表示し，それがゆっくり画面上を移動し，画面の枠で跳ね返る様子を表現．

2 ピンポン球の表示

本誌2007年8月号特集1の第2章でVGA画面表示の同期信号を作りました．まず**図1**に示すように10ドット×10ドットのピンポン球（画面上では正方形）を表示してみましょう．次にこのピンポン球を画面上をゆっくり移動させて画面の枠で跳ね返る様子を表現します．

リスト1に示すように，

- ピンポン球の水平方向位置を示すh_pos
- 垂直方向位置を示すv_pos

を定義します．初期値としてピンポン球の左上角の座標を（144，244）としました．**リスト2**の記述を追加するとピンポン球が画面の中ほどに表示されます．

リスト1 ピンポン玉の表示（内部信号の定義）

```
signal h_pos      : std_logic_vector (9 downto 0)
                          :="0010010000"; --144
signal v_pos      : std_logic_vector (9 downto 0)
                          :="0011110100"; --244
```

リスト2 ピンポン玉の表示（画面の中ほどにピンポン玉を白色で表示．ただし四角の球？）

```
rgb<="111" when h_counter>=h_pos and
                        h_counter<h_pos+10 and
             v_counter>=v_pos and v_counter<v_pos+10
else
     "000";
LR<=(others=>rgb(2));
LG<=(others=>rgb(1));
LB<=(others=>rgb(0));
```

KeyWord ADuC7026，XC3S500E-VQ208，Spartan-3E，VGA，RGB，CQ-SP3EDW，CQ-SP3E208，画像フレーム・メモリ，LDO，ブロック崩しゲーム

3 ピンポン球の移動

次にリスト3に示すように，三つの内部信号を定義します．

- クロック　puls
- ピンポン球の水平方向の動き　h_dir
- ピンポン球の垂直方向の動き　v_dir

pulsは，垂直カウンタの一部から取り出した適度な速さのクロックです．h_dir，v_dirはピンポン球の移動方向を示します．pulsの立ち上がり時に，ピンポン球の移動方向に位置レジスタの値を1ずつ加減します．これによってピンポン球が移動します．移動の結果，ピンポン球の位置が画面の端に到達した場合はピンポン球の移動方向を逆にします．

リスト4は水平方向の動きに対する制御の記述です．同様に垂直方向の制御も追加すると，ピンポン球は図2に示すように画面の枠内を移動します．

4 パドルの追加

さて次は図3のように画面の中央下部分にパドル(サイズ80ドット×8ドット)を配置します．パドルに当たるとピンポン球が跳ね返るように記述を変更します．

リスト5はパドルの属性定義で，パドルの左上角の座標を(280，384)としています．パドルのサイズは固定値(constant)，位置座標は変化可能なレジスタ値(signal)として定義します．

リスト6はパドルの配置とピンポン球の跳ね返りの記述です．パドルの上側だけでなく，下側でもピンポン球が跳ね返るよう，2カ所に分けて記述してあります．

図2 ピンポン球の動作(上下左右に移動)
画面の上下左右枠で跳ね返りながら，斜め上下の移動を繰り返す．

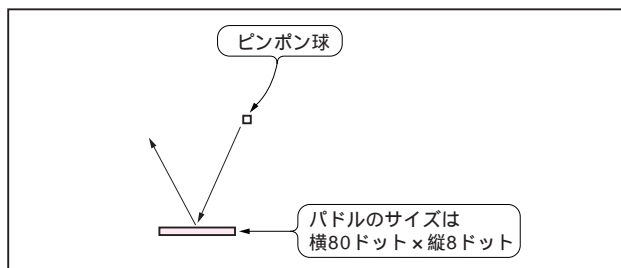
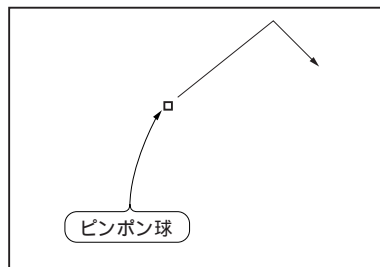



図3 パドルの配置
画面の中央下部分にパドル(サイズ80ドット×8ドット)を配置し，その部分でもピンポン球が跳ね返るように変更．

5 ブロック崩しの記述

今度はピンポン球の跳ね返りを利用して，図4のように配置したブロックを消していく「ブロック崩し」にしてみましょう．

まずブロックを配置します．ブロックの有無はリスト7に示すようにレジスタblk1のビットで示します(1：あり，0：なし)．

リスト3 動きのあるピンポン玉の表示(内部信号定義)
クロックとしてpuls，ピンポン玉の動きの方向は，水平方向を示すh_dir，垂直方向を示すv_dirを定義

```
signal h_dir   : std_logic :='0';   ･･･ 0:左へ  1:右へ
signal v_dir   : std_logic :='0';   ･･･ 0:下へ  1:上へ
signal puls    : std_logic;
```

リスト4 動きのあるピンポン玉の表示(水平方向のみ制御)
ピンポン玉移動用のクロックの生成と，水平方向の動きに対する制御部分．

```
puls<=v_counter(6);

process (puls)
begin
      if puls'event and puls='1' then
            if h_dir='1' then
                  h_pos<=h_pos+1;
                  if h_pos>630 then
                        h_dir<='0';
                  end if;
            else
                  h_pos<=h_pos-1;
                  if h_pos<=1 then
                        h_dir<='1';
                  end if;
            end if;

      end if;
end process;
```

リスト5 パドルの配置(内部信号定義の追加)
パドルの左上角の座標を(280，384)とする．パドルのサイズは固定(constant)で位置座標は変化できるよう定義．

```
signal pad_h_pos     : std_logic_vector (9 downto 0)
                            :="0100011000"; --280
signal pad_v_pos     : std_logic_vector (9 downto 0)
                            :="0110000000"; --384
constant pad_h_size : std_logic_vector (9 downto 0)
                             :="0001010000"; --80
constant pad_v_size : std_logic_vector (9 downto 0)
                              :="0000001000"; --8
```

3

リスト6　パドルの配置と跳ね返りの制御

垂直制御部分に，パドルの表示とパドルでの跳ね返り制御を加えたもの．パドルの上側だけでなく，下側でもピンポン球が跳ね返るように，2箇所に分けて記述．

```
puls<=v_counter(6);

process (puls)
begin
        if puls'event and puls='1' then

        ......................    (水平方向制御)

                if v_dir='1' then
                        v_pos<=v_pos+1;                 パドル上側跳ね返り制御
                                if v_pos+10=pad_v_pos and h_pos>=pad_h_pos and h_pos<pad_h_pos+pad_h_size then
                                        v_dir<='0';
                                end if;

                        if v_pos>470 then
                                v_dir<='0';
                        end if;
                else
                        v_pos<=v_pos-1;                 パドル下側跳ね返り制御
                                if v_pos=pad_v_pos+pad_v_size and h_pos>=pad_h_pos and h_pos<pad_h_pos+pad_h_size then
                                        v_dir<='1';
                                end if;


                        if v_pos<=1 then
                                v_dir<='1';
                        end if;
                end if;

        end if;
end process;

rgb<=
 "111" when h_counter>h_pos and h_counter<h_pos+10
             and v_counter>v_pos and v_counter<v_pos+10 else

 "011" when h_counter>=pad_h_pos and h_counter<pad_h_pos+pad_h_size
           and v_counter>=pad_v_pos and v_counter<pad_v_pos+pad_v_size else     パドルを
                                                                                 水色で表示
 "000";
```

リスト8　ブロックの赤色表示

```
rgb<=
 "111" when h_counter>=h_pos and h_counter<h_pos+10
           and v_counter>=v_pos and v_counter<v_pos+10 else

 "011" when h_counter>=pad_h_pos and h_counter<pad_h_pos+pad_h_size
           and v_counter>=pad_v_pos and v_counter<pad_v_pos+pad_v_size else
 "100" when blk1(0)='1' and h_counter>0 and h_counter<0+30 and v_counter<12 else
 "100" when blk1(1)='1' and h_counter>32 and h_counter<32+30 and v_counter<12 else
 "100" when blk1(2)='1' and h_counter>64 and h_counter<64+30 and v_counter<12 else
 "100" when blk1(3)='1' and h_counter>96 and h_counter<96+30 and v_counter<12 else
 "100" when blk1(4)='1' and h_counter>128 and h_counter<128+30 and v_counter<12 else
 "100" when blk1(5)='1' and h_counter>160 and h_counter<160+30 and v_counter<12 else
 "100" when blk1(6)='1' and h_counter>192 and h_counter<192+30 and v_counter<12 else
 "100" when blk1(7)='1' and h_counter>224 and h_counter<224+30 and v_counter<12 else
 "100" when blk1(8)='1' and h_counter>256 and h_counter<256+30 and v_counter<12 else
 "100" when blk1(9)='1' and h_counter>288 and h_counter<288+30 and v_counter<12 else
 "100" when blk1(10)='1' and h_counter>320 and h_counter<320+30 and v_counter<12 else
 "100" when blk1(11)='1' and h_counter>352 and h_counter<352+30 and v_counter<12 else
 "100" when blk1(12)='1' and h_counter>384 and h_counter<384+30 and v_counter<12 else
 "100" when blk1(13)='1' and h_counter>416 and h_counter<416+30 and v_counter<12 else
 "100" when blk1(14)='1' and h_counter>448 and h_counter<448+30 and v_counter<12 else
 "100" when blk1(15)='1' and h_counter>480 and h_counter<480+30 and v_counter<12 else
 "100" when blk1(16)='1' and h_counter>512 and h_counter<512+30 and v_counter<12 else
 "100" when blk1(17)='1' and h_counter>544 and h_counter<544+30 and v_counter<12 else
 "100" when blk1(18)='1' and h_counter>576 and h_counter<576+30 and v_counter<12 else
 "100" when blk1(19)='1' and h_counter>608 and h_counter<608+30 and v_counter<12 else


 "000";
```

初期値としてレジスタblk1の各ビットはすべて'1'にセットし，20個のブロックを表示します．

リスト8はブロックの表示記述です．ブロックは赤色で表示します．ピンポン球が当たったブロックの消去は**リスト8**の水色の部分で実現できます．

リスト9はパドルの配置とパドルに当たったピンポン球の跳ね返りの記述です．制御ブロックの下端の垂直座標値は11です．

垂直カウンタがその値に達したとき，ブロックを端から順に調べていきます．該当のブロックが存在し，ピンポン球の水平位置がそのブロックの位置と一致した場合はblk1の該当ビットをリセットします．そしてピンポン球の垂直移動方向を下向きに変えます．

ブロック崩しゲームのVHDLソース・リストは付属CD-ROMに収めました．

いかがでしょうか．パドルは固定ですがブロックが徐々に消されていく様子が観察できます．**写真1**は完成したデモ画面です．ブロックを3列にして消されたブロックの数をカウントし，2けたの7セグメント数字で表示しています．

ブロックの大きさ
横30ドット×縦12ドット

図4　ブロック崩し
ピンポン球の跳ね返りを利用して配置したブロックを消していく．

6　仕上げはパドル操作 …ゲームとして完成！

ゲームとして完成させるためには「プレーヤの操作によってパドルを左右に移動させる」仕組みが必要です．**図5**に示すようにスライド・ボリューム(可変抵抗器)を画像ベースボードのアナログ入力電圧端子ADC0に接続します．

写真2に示すようにスライド・ボリュームを接続し，端子を左右にスライドさせるとADC0の電圧が変化します．

リスト7　パドルの配置(内部信号定義追加)

```
signal blk1    : std_logic_vector (19 downto 0)
               :="11111111111111111111";
```

リスト9　パドルの配置と跳ね返りの制御

```
puls<=v_counter(6);

process (puls)
begin
        if puls'event and puls='1' then

      ......................    (水平方向制御)

                if v_dir='1' then
                        ................
                else
                        v_pos<=v_pos-1;

                        if v_pos=pad_v_pos+pad_v_size and h_pos>=pad_h_pos and h_pos<pad_h_pos+pad_h_size then
                                v_dir<='1';
                        end if;
                                if v_pos=11 then
                                        for j in 0 to 19 loop
                                            if( blk1(j)='1' and  h_pos+5>=j*32 and h_pos+5<j*32+30) then
                                                        blk1(j)<='0';
                                                        v_dir<='1';
                                                end if;
                                        end loop;
                                end if;


                        if v_pos<=1 then
                                v_dir<='1';
                        end if;
                end if;

        end if;
end process;
```

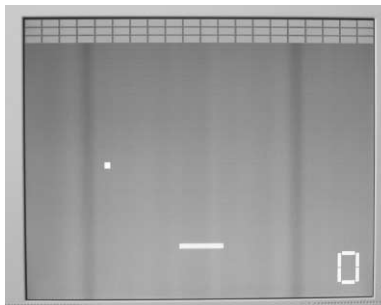



写真1　ブロック崩しゲームの画面

完成したデモ画面．パドルは動かないが，ブロックが徐々に消されていく様子が観察できる．3列のブロックの消された数をカウントし，2けたの7セグメント数字で表示する．

図5
スライド・ボリュームによるパドル位置入力

スライドした位置をもとにアナログ電圧を生成する．アナログ電圧に応じてパドルが左右する．

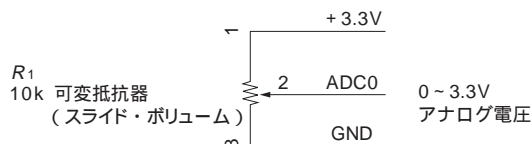
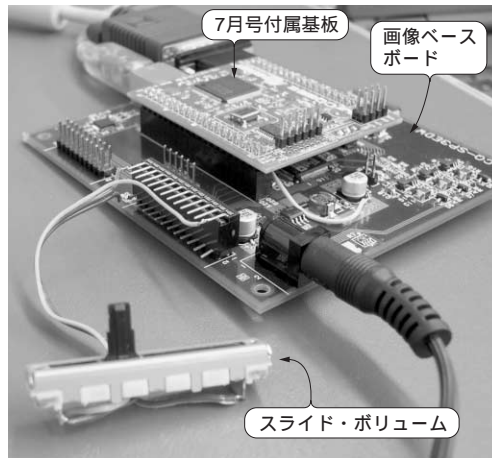



写真2
スライド・ボリュームでパドル位置を入力する

本誌2007年7月号の付属基板と画像ベースボードで構成したブロック崩しゲーム．画像ベースボードのA-D変換入力端子にスライド・ボリュームを接続する．ボリューム端子を左右にスライドさせると，ADuC7026のADC0端子電圧が変化する．これを読み込んでFPGA内のレジスタに引き渡す．

リスト10　ラケット・データ入力プログラム

スライド・ボリュームの電圧をA-D変換し，その値をFPGA内のラケット位置レジスタに書き込むプログラム．

```
/*********************************************
   ラケット位置入力
**********************************************/
#include <ADuC7026.H>       // ADuC7026 レジスタ定義ファイル

#define RC     (*(volatile unsigned short *)
0X10000000)    //ラケット位置レジスタ
void  AD_init(void);
short ADIN(int num);

/*********************************************
    A/D変換器   （初期化＆変換入力）
 *********************************************/
void AD_init(void)
{
      int i = 0;
      ADCCON = 0x0020;              // power-on the ADC
      while (i <= 20000) {          // wait for ADC to be
                                           fully powered on
        i++;
      }
      REFCON = 0x03;                // internal 2.5V
                                reference. 2.5V on Vref pin
      ADCCON = 0x17A4;              // ADC Config:
       fADC/32, acq. time = 16 clocks => ADC Speed = 1MSPS
}

short ADIN(int num)
{
      short data;
      ADCCP = num;
      while (!ADCSTA);        // wait for end of conversion
      data = ADCDAT >> 16;
      data &= 0x0FFF;
      return data;
}

int main (void){
      // PLL
      PLLKEY1   = 0x000000aa;  // Release PLLKEY
      PLLCON    = 0x00000021;  // Uset external 32kHz
      PLLKEY2   = 0x00000055;  // Set PLLKEY
      // Clock select
      POWKEY1   = 0x00000001;  // Release POWKEY
      POWCON    = 0x00000000;  // Active mode,41.78MHz
      POWKEY2   = 0x000000f4;  // Set POWKEY
      // LED
      GP0CON    = 0x00000000;
      GP0PAR    = 0x20000000;
      GP1DAT    = 0xFF000000;  // P1.7 Output '0'
      GP1DAT    = 0xFFE40000;
      // External memory
      GP2CON    = 0x00022220;  // nMS0,AE,nRD,nWR
      GP3CON    = 0x22222222;  // AD7-AD0
      GP4CON    = 0x22222222;  // AD15-AD8
      XMCFG     = 0x00000001;  // External memory enable
      XM0CON    = 0x00000003;  // 16bit bus, XM0 enable
      XM0PAR    = 0x00008710;  // WR enable, nWR strobe 2CLK

      AD_init();

       while (1){
             RC=ADIN(0)*599/4095;        // 0 cnannel
    }
}
```

リスト10はADuC7026のパドル・データ入力プログラムです．スライド・ボリュームの電圧をA-D変換し，その値をFPGA内部のレジスタRCに書き込みます．

FPGA内部のロジックはレジスタRCの値をパドルの位置情報として受け取ります．

参考・引用*文献

(1) アナログ・デバイセズ；高精度アナログ・マイクロコントローラ ADuC7019/20/21/22/24/25/26/27データシート，2006年.

えさき・まさやす
(株)イーエスピー企画　代表取締役
いわた・まさお